Cap. 5 Ricostruzione delle tracce con algoritmi genetici

5.1 Introduzione agli Algoritmi Genetici

"Gli organismi viventi sono esperti solutori di problemi"; questa osservazione di John Henry Holland, l'inventore degli Algoritmi Genetici, è all'origine dei tentativi di emulare la versatilità e la potenza dell'evoluzione naturale per scopi informatici; i primi lavori risalenti agli anni '50 e '60 erano incentrati sulla simulazione dell'evoluzione naturale (Barricelli nel 1954 simulava l'evoluzione di automi in grado di misurarsi con un semplice gioco di carte); in quel periodo maturò l'idea che l'evoluzione "naturale" poteva essere un buon metodo di ottimizzazione. Fu però con Holland, e con i suoi lavori dei primi anni '70 che fu coniato il termine algoritmi genetici, che salirono agli onori della cronaca nel 1975 con il suo libro "Adaptation in Natural and Artificial Systems" [46].

Gli organismi viventi evolvono attraverso la selezione naturale e la riproduzione (nella maggior parte dei casi sessuata), a cui si aggiunge una possibile mutazione casuale (dai possibili tristi effetti nella nostra specie), che però ha il pregevole compito di salvaguardare intere specie dal pericoloso effetto di ristagno dato dall'involuzione. In natura l'individuo che trasmette alle generazioni successive il proprio patrimonio genetico è quello che riesce a sopravvivere più a lungo e che riesce a vincere le competizioni per aggiudicarsi il diritto ad accoppiarsi, in una parola il più "adatto". I figli hanno un'alta probabilità di ereditare le buone caratteristiche dei genitori, anche dopo il parziale rimescolamento del materiale genetico tra madre e padre, anzi c'è una probabilità non trascurabile che uno dei figli (più sono e meglio è) erediti le caratteristiche migliori di entrambi i genitori; i figli peggiori saranno destinati nel migliore dei casi ad essere passivi spettatori della storia dell'evoluzione, o altrimenti a soccombere.

Questa caratteristica dell'evoluzione verso il miglior adattamento (fitness) è stata sfruttata per implementare un algoritmo di ottimizzazione simulando una popolazione di individui, una funzione di fitness (che è la funzione da massimizzare), un cross-over tra gli individui, e una mutazione. Ogni individuo rappresenta un punto nello spazio delle possibili soluzioni e durante l'algoritmo la popolazione evolve verso punti via via migliori (con fitness più elevata).

Un primo problema da risolvere è quello della codifica dell'informazione nei "geni": "mutando" o "accoppiando" il testo di un programma FORTRAN non si ottiene un programma FORTRAN migliore o peggiore di quello iniziale, ma di solito si ottiene un errore di compilazione... Bisogna che il meccanismo del cross-over e della mutazione diano risultati accettabili, anche se in qualche caso peggiori di quelli di partenza.

Bisogna pertanto individuare le variabili significative del problema e una loro funzione da massimizzare; questo potrebbe già essere un compito arduo, tuttavia questa funzione può essere molto generica, non ci sono particolari vincoli di regolarità, anzi la funzione può anche non essere definita in modo algebrico.

I parametri rispetto a cui ottimizzare la funzione vanno codificati, opportunamente quantizzati, in stringhe binarie. In questo modo una funzione di due parametri reali diciamo compresi tra 0 e 1:

$$x, y \in [0,1]$$

quantizzati a 16 bit (da 0 a 65535) sarà ricercata in una "popolazione " di cromosomi binari di lunghezza 32. Il numero di possibili soluzioni è quindi 2^{32}, pari a circa 4 miliardi di combinazioni.

La cosa più importante negli algoritmi genetici è sicuramente la funzione di "Fitness" che è il vero legame con la realtà che si vuole simulare e ottimizzare. L'algoritmo genetico infatti ricercherà il massimo (o se si vuole il minimo, e allora invece di fitness si potrà chiamare funzione costo o errore) di questa funzione.

La caratteristica che rende gli algoritmi genetici molto interessanti è che la ricerca del minimo avviene in modo "globale", cioè non c'è il pericolo che la soluzione si "intrappoli" in un minimo locale della funzione, come invece avviene per gli usuali metodi detti "Steepest Descent" (o "Hill Climbing" a seconda del verso della funzione). Appare chiaro quindi che l'utilità sarà massima quando la funzione è molto complessa, è fortemente non lineare, ha molti massimi e minimi relativi, non è facilmente descrivibile in termini matematici.

Un'altra caratteristica è il "parallelismo intrinseco" dovuto all'esistenza di una "popolazione" di possibili soluzioni che esplorano lo spazio in modo caotico (o meglio semi-caotico); questa caratteristica è comune ad esempio agli algoritmi randomici, ma si vedrà che l'operazione di accoppiamento (cross-over) degli individui migliori (selezione) indirizza l'esplorazione dello spazio in modo ottimo (o quasi, ma comunque in modo più efficiente della ricerca randomica).

L'algoritmo comincia con una "popolazione" iniziale di elementi random. Ad ogni iterazione (generazione) viene valutata la fitness di ogni elemento.

Vengono selezionati gli elementi migliori per l'accoppiamento (cross-over); di solito si utilizza un criterio di Selezione Proporzionata alla fitness (Roulette Wheel Selection), in base al quale un individuo x_j è scelto come genitore con probabilità

$$P_{SEL} = \frac{fit(x_j)}{\sum_i fit(x_i)}$$

Per ogni coppia di genitori selezionati si applica il cross-over con una certa probabilità P_C preassegnata (ad es. 80%).

Vi sono vari tipi di cross-over; il più semplice è quello con un punto di taglio: data una coppia di genitori viene scelto in modo casuale il punto di taglio (da 1 ad n-2 per cromosomi di n bit), e vengono scambiate le sequenze di bit successive al taglio (vedi Figura 79)

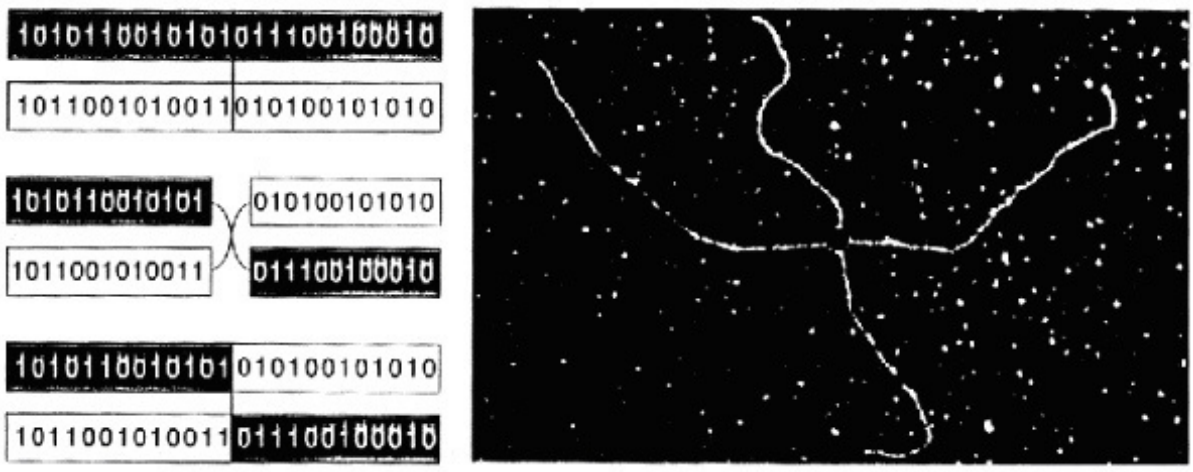

Figura 79 - Schematizzazione binaria e fotografia del momento del cross-over in cromosomi reali.

In questo modo i genitori trasferiscono ai figli parte del loro patrimonio genetico.

In seguito viene applicata la mutazione (cioè l'inversione di un bit random) con probabilità P_{Mut} di solito dell'ordine del 1‰. Questo ha lo scopo di evitare in caso di convergenza evolutiva, il ristagno della popolazione.

A questo punto si riparte con la valutazione della fitness per i nuovi individui e così via, fino ad un criterio di terminazione dell'algoritmo, che può essere basato sul valore del massimo della fitness per ogni generazione e/o sul numero di generazioni.

Il diagramma di flusso dell'algoritmo è riportato in Figura 80.

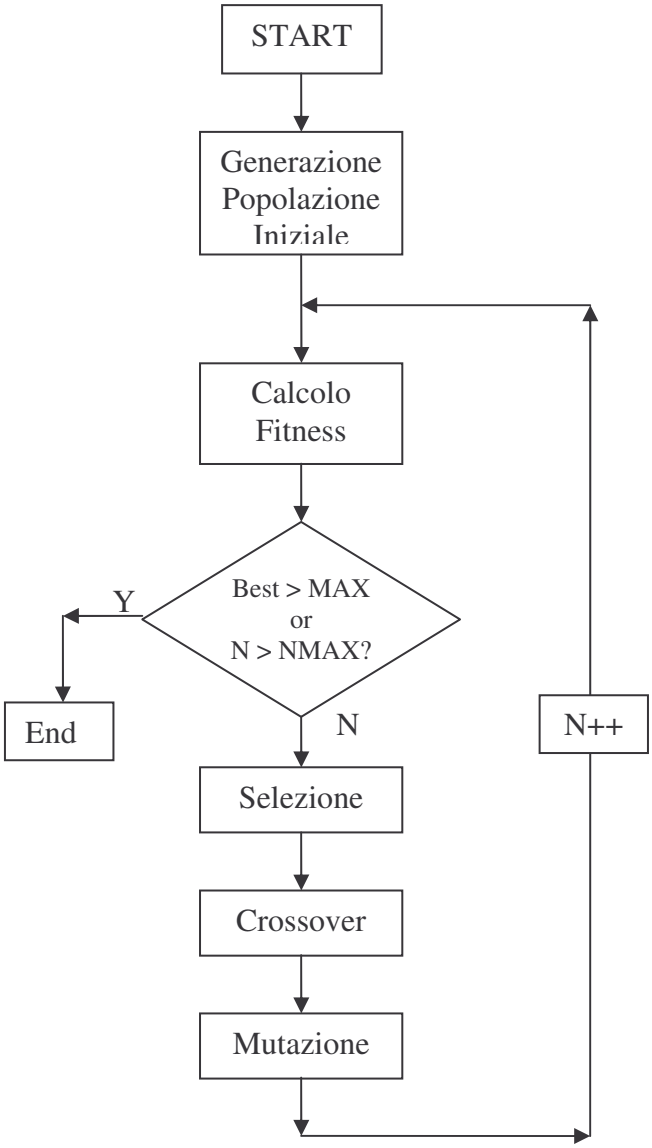

Figura 80 - Diagramma di flusso di un algoritmo genetico.

5.1.1 Teoria: perché funziona? Iperpiani negli spazi di ricerca.

La domanda che molti si pongono a questo punto è: perché un tale processo dovrebbe produrre qualcosa di utile? Perché tutto questo dovrebbe portare ad un'efficace forma di ottimizzazione?

La risposta che di solito viene data per spiegare il comportamento degli algoritmi genetici è il risultato del lavoro di Holland, che nel suo già citato libro [46] sviluppa vari argomenti per spiegare come un algoritmo genetico può fornire una ricerca complessa e robusta andando a campionare partizioni iper-planari dello spazio di ricerca.

Il miglior modo di capire come un algoritmo genetico può campionare iperpiani dello spazio di ricerca è di considerare un semplice spazio 3-dimensionale (vedi Figura 81). Assumendo di avere un problema codificato con soli 3 bit, lo spazio di ricerca può essere rappresentato da un cubo (la stringa 000 coincide con l'origine degli assi). Gli angoli del cubo sono individuati da tutte le possibili stringhe di 3 bit, e angoli adiacenti hanno stringhe che differiscono di un solo bit (vedi Figura 81). Il piano frontale del cubo contiene tutti i punti che iniziano per 0. Utilizzando il simbolo * come jolly questo piano può essere indicato dalla stringa 0**. Le stringhe contenenti * sono dette schemi[7].

Ogni schema corrisponde ad un iperpiano nello spazio di ricerca. L'ordine di un iperpiano è il numero di bit indicati effettivamente nello schema: quindi ad esempio 0** è uno schema di ordine 1, mentre 1**1******0* è di ordine 3.

Per rappresentare uno spazio di ricerca 4-dimensionale si può inserire un cubo nel cubo precedente, come in basso nella Figura 81. I punti possono essere indicati come per il cubo originario, sia quelli appartenenti al cubo esterno che quelli del cubo interno con le stringhe di 3 bit precedenti; a questo punto basta aggiungere all'inizio delle stringhe uno 0 per i punti del cubo esterno e un 1 per i punti del cubo interno. Il cubo interno ora è individuato dallo schema 1*** e quello esterno dallo schema 0***. Analogamente è facile vedere che lo schema *0** indica i punti appartenenti ai piani frontali di entrambi i cubi, e 10** è il piano frontale del cubo interno.

[7] schema o al plurale schemata dal greco σχήμα, pl. σχήματα che vuol dire forma, piano

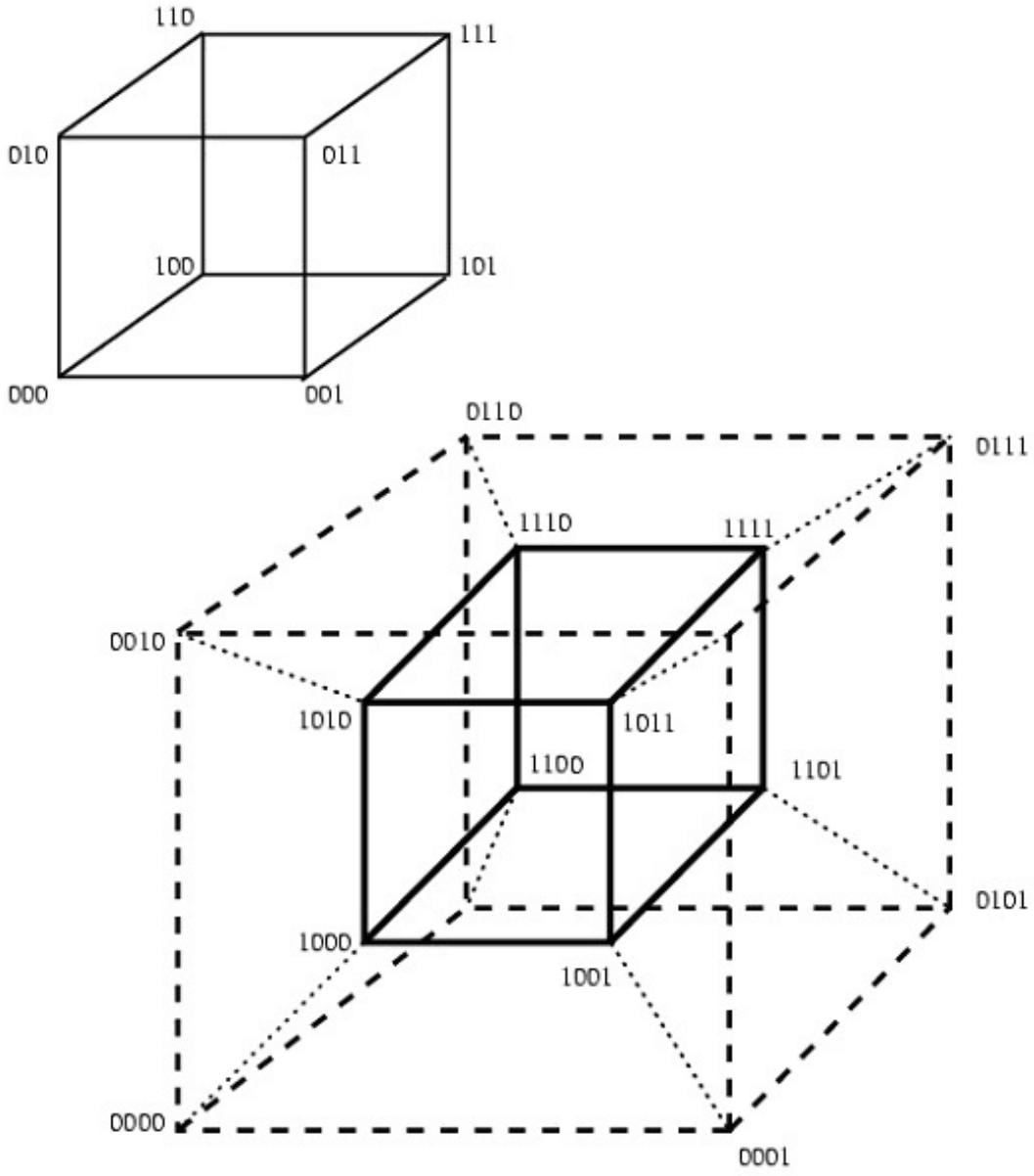

Figura 81 - Cubo e ipercubo di ordine 4 [45].

Ogni codice binario è un "cromosoma" che corrisponde ad un punto dello spazio di ricerca. Ogni punto è inoltre membro di 2^L-1 iperpiani: sono i modi in cui posso disporre degli * in una stringa di L bit data. Il numero totale di iperpiani nello spazio di ricerca è invece 3^L-1: in questo caso ogni bit può assumere il valore 0, 1 o *. L'iperpiano banale coincidente con l'intero spazio di ricerca non è considerato, mentre sono considerati come iperpiani anche i singoli punti (iperpiani di ordine L).

Il parallelismo intrinseco in un algoritmo genetico non è solamente dato dal fatto che abbiamo una popolazione di possibili soluzioni, ma anche dal fatto che ogni cromosoma in realtà campiona un gran numero di iperpiani dello spazio di ricerca. L'effetto cumulativo di N iperpiani per M individui della popolazione fornisce una informazione statistica su tutti i sottoinsiemi di iperpiani [46].

Il parallelismo intrinseco implica che molte competizioni tra iperpiani vengano risolte simultaneamente in parallelo. La teoria suggerisce che attraverso il processo di riproduzione e ricombinazione, gli schemi aumentino o diminuiscano la loro rappresentanza nella popolazione a seconda della fitness delle stringhe che giacciono in quegli iperpiani.

Un altro modo di considerare le partizioni a iperpiano è presentato in Figura 82. Viene disegnata la funzione fitness da massimizzare in funzione di una variabile di uno spazio unidimensionale. L'iperpiano 0***..** ricopre la prima metà dello spazio, e 1***...** ricopre la seconda metà. Poiché le stringhe dell'iperpiano 0***...** sono in media migliori di quelle dell'iperpiano 1***...**, si vorrebbe che la ricerca fosse indirizzata preferibilmente sulla prima partizione. Nella seconda parte della figura è evidenziata la partizione **1**...** e si nota anche la sovrapposizione tra 0***...** e **1**...**, cioè l'intersezione 0*1***...**. Quindi nel terzo grafico si può osservare la partizione 0*10**...**.

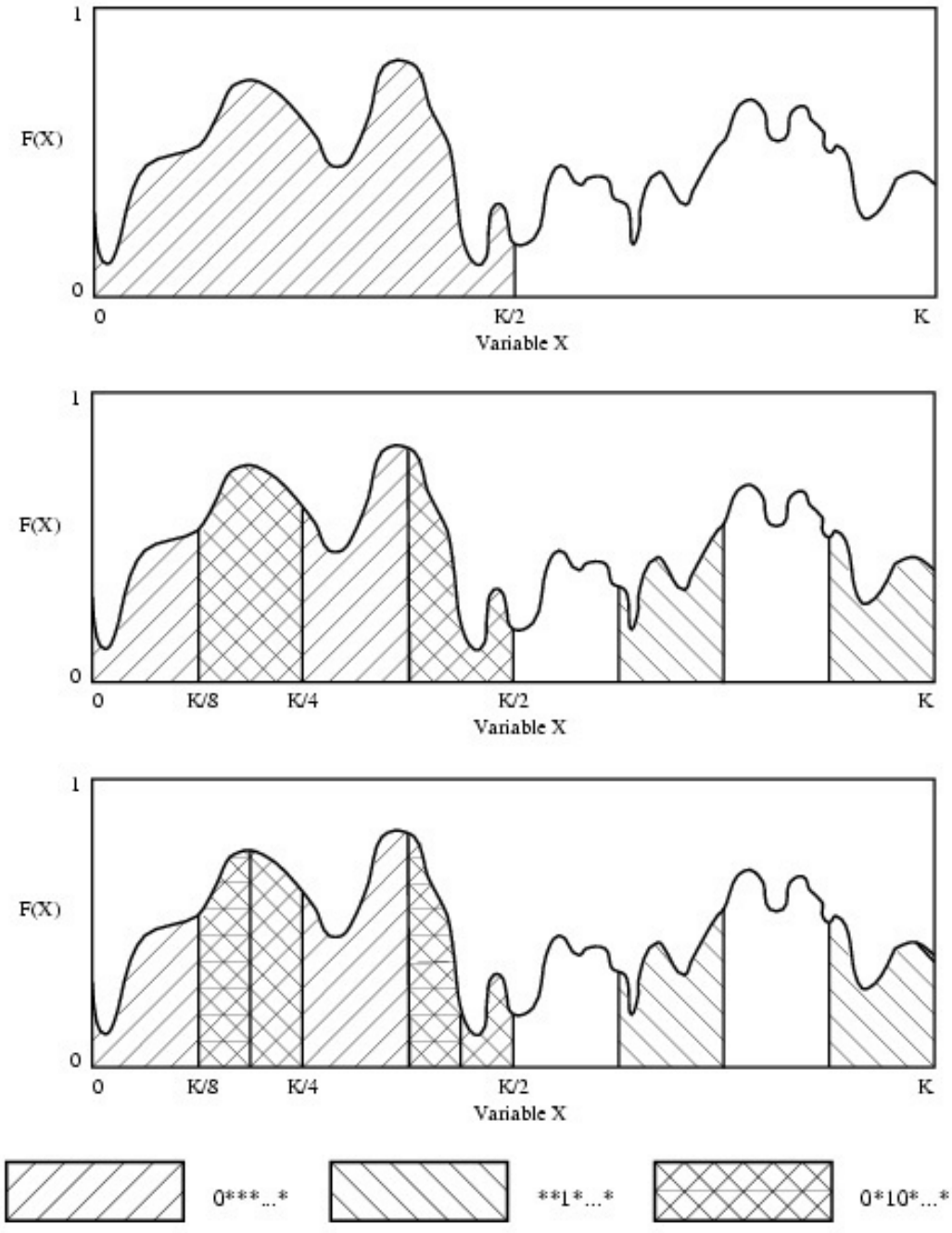

Figura 82 - Rappresentazione degli iperpiani nel dominio della funzione fitness [45].

Dalla Figura 82 si capisce che il campionamento delle partizioni non è condizionato da massimi locali. Aumentando il tasso di campionamento di partizioni che hanno valori di fitness superiori alla media aumenta la probabilità di trovare soluzioni ottimali.

Indicando con $M(H,t)$ il numero di stringhe di bit che appartengono ad un certo iperpiano H alla t-esima generazione, è possibile trovare un limite inferiore a $M(H,t+1)$

(Schemata theorem). Cioè, è possibile dimostrare che il campionamento delle soluzioni si infittisce sugli iperpiani che presentano una fitness superiore alla media.

Intanto indichiamo con $M(H,t+1/2)$ il numero di stringhe appartenenti ad H dopo la selezione, ma prima del cross-over e della mutazione. Eseguendo la selezione proporzionale si avrà

$$M\left(H,t+\frac{1}{2}\right)=M\left(H,t\right)\frac{f\left(H,t\right)}{\bar{f}} \tag{9}$$

dove $f(H,t)$ è la fitness dell'iperpiano H, e $\bar{f}$ è la fitness media su tutta la popolazione.

Vediamo come il cross-over e la mutazione intervengono a modificare la (9).

Operatori di Cross-Over e Schemi

Osserviamo innanzi tutto che gli iperpiani di ordine 1 non vengono modificati dall'operatore di cross-over, infatti l'unico bit del relativo schema di ognuno dei genitori viene ereditato da uno dei due discendenti. Invece la distribuzione dei campioni degli iperpiani di ordine superiore al primo può essere alterata dal cross-over. Inoltre iperpiani dello stesso ordine possono essere alterati con probabilità diverse.

Cross-over ad 1 punto di taglio

Questo operatore è comodo perché è abbastanza semplice quantificarne l'effetto sui vari schemi. Assumendo di lavorare su stringhe di 12 bit consideriamo i seguenti schemi: 11********** e 1**********1. La probabilità che i due bit del primo schema vengano separati dal cross-over è soltanto

$$\frac{1}{L-1}$$

perché ci sono L-1 punti di taglio nel cross-over a 1 punto. Al contrario la probabilità che il secondo schema venga alterato è unitaria, perché ogni punto di taglio separa i due bit dello schema. Per il cross-over ad un punto di taglio quindi la posizione dei bit dello schema è importante per determinare la probabilità che quei bit rimangano uniti dopo il cross-over, cioè che lo schema non venga alterato.

Cross-Over a 2 punti di taglio

Questo tipo di cross-over usa due punti scelti a caso; le stringhe si scambiano il segmento che cade tra i due punti. Si osservi che il cross-over a 2 tagli tratta le stringhe e gli schemi come se fossero chiusi a formare un anello, come illustrato nella Figura 83

Figura 83 - Schemi nel cross-over a 2 tagli.

dove b1 .. b12 rappresentano i bit da 1 a 12. Quando visto in questo modo, il cross-over ad un punto è un caso particolare di quello a 2 punti: dove cioè uno dei due punti capita sempre nella posizione di ricongiungimento tra il primo e l'ultimo bit. Ora il massimo disturbo per gli schemi di ordine 2 si ha quando i due bit occupano posizioni complementari sull'anello. Quindi gli schemi che hanno bit vicini tra loro hanno meno probabilità di essere scompaginati dal cross-over ad uno o a 2 tagli. Più precisamente gli iperpiani rappresentati da schemi compatti saranno campionati con frequenza più vicina a quella che si avrebbe con la sola selezione, minimizzando così la probabilità che il cross-over ne disturbi la distribuzione.

Ogni iperpiano o schema è quindi caratterizzato dalla sua "Lunghezza", cioè dalla distanza tra il primo e l'ultimo bit non jolly (0 o 1), indichiamola con $\Delta(H)$. Maggiore è la lunghezza caratteristica di un iperpiano, maggiore sarà la probabilità che esso venga distrutto durante il Cross-Over. In particolare, per il Cross-Over ad un taglio si ha che la probabilità di distruzione (p_D) dello schema che rappresenta l'iperpiano H è:

$$p_D \propto \frac{\Delta(H)}{L-1}$$

In realtà un punto di taglio interno alla lunghezza caratteristica di uno schema non è detto che causi una distruzione dello schema stesso: infatti, se due sottosequenze omologhe dei genitori sono uguali, lo schema non viene interessato dal Cross-Over, questo avviene sicuramente se entrambi i genitori appartengono allo schema. Dato un elemento appartenente allo schema H,

la probabilità che un partner random appartenga allo stesso schema è *P(H,t),* pari a *M(H,t)* diviso la dimensione della popolazione;

$$p_D \cong \frac{\Delta(H)}{L-1}\left[1-P(H,t)\right] \tag{10}$$

Il teorema degli Schemi

Abbiamo visto l'effetto della selezione sulla popolazione di stringhe appartenenti ad un dato iperpiano, ora possiamo includere l'effetto del Cross-Over, per arrivare al cosiddetto teorema degli schemi [46].

Innanzi tutto bisogna considerare che il Cross-Over è applicato con una probabilità minore di 1, quindi una parte della popolazione rimane inalterata, e questo dà luogo al primo termine pesato per ($1-p_C$). La parte di popolazione che subisce il Cross-Over verrà alterata, e ci saranno delle stringhe che usciranno dallo schema (perdite), e altre che entreranno nello schema (guadagni). Le perdite sono valutabili attraverso la "distruzione" dello schema data dall'applicazione dell'operatore di Cross-Over (10), mentre i guadagni sono dovuti alla comparsa in uno schema di una stringa proveniente da genitori fuori dallo schema stesso; un esempio è dato dallo schema 11******, e dai genitori 10****** e 01******, con il Cross-Over ad un taglio tra il primo e il secondo bit: uno dei figli entra nello schema dato. Dopo il Cross-Over il numero di stringhe *M(H, t+1)* sarà:

$$M(H,t+1)=(1-p_C)M(H,t)\frac{f(H,t)}{\bar{f}}+p_C\left[M(H,t)\frac{f(H,t)}{\bar{f}}(1-perdite)+guadagni\right]$$

Facendo un'ipotesi semplificativa, ma conservativa, non consideriamo ulteriormente i possibili guadagni, e consideriamo le perdite uguali alla probabilità di distruzione massima data dalla (10). A parte il caso fortunato in cui entrambi i genitori appartengano allo schema, si ha quindi:

$$M(H,t+1)\geq(1-p_C)M(H,t)\frac{f(H,t)}{\bar{f}}+p_C\left[M(H,t)\frac{f(H,t)}{\bar{f}}\left(1-\frac{\Delta(H)}{L-1}\left(1-P(H,t)\ \right)\right)\right] \tag{11}$$

Dividendo per la dimensione della popolazione e riarrangiando, si ha il teorema degli schemi, cioè il limite inferiore per l'evoluzione della rappresentatività nella popolazione al tempo t+1 dell'iperpiano *H*:

$$P(H,t+1)\geq P(H,t)\frac{f(H,t)}{\bar{f}}\left[1-p_C\,\frac{\Delta(H)}{L-1}\left(1-P(H,t)\frac{f(H,t)}{\bar{f}}\right)\right] \tag{12}$$

Dalla (9) e dalle successive correzioni, si vede che gli individui con valori di fitness più elevata hanno maggiore probabilità di sopravvivere.
Quindi gli schemi con una lunghezza caratteristica non troppo elevata e con una fitness superiore alla media avranno una rappresentatività esponenzialmente crescente. In effetti questo è vero per qualche generazione, poi il bias introdotto nella distribuzione degli individui insieme alle approssimazioni fatte per giungere alla (11) non permette di valutare esattamente l'andamento nel tempo della rappresentatività. La (12) dà pertanto un'indicazione della convergenza della popolazione in media verso soluzioni migliori.Cross-Over, Mutazione e convergenza prematura

Dal teorema degli schemi sembra che il Cross-Over crei solo svantaggi, a causa della probabilità di distruggere uno schema con una buona fitness; questo tuttavia è solo un aspetto del complesso problema dell'evoluzione di un Algoritmo Genetico; infatti, se fosse attiva solo la selezione, con poche generazioni tutti gli individui apparterrebbero allo schema migliore presente al momento dell'inizializzazione, ma si assisterebbe ad una cosiddetta convergenza evolutiva. Per evitare questo impasse si utilizza l'operatore di Mutazione, che cambia dei bit random con probabilità p_{MUT}.

Altri meccanismi euristici per favorire l'evoluzione sono legati invece alla capacità di "esplorazione" dell'operatore di Cross-Over: il Cross-Over ad un taglio, dai genitori può produrre al massimo solo $2(L\text{-}1)$ figli diversi, mentre il Cross-Over a due tagli può generare $2\binom{L}{2} = L^2 - L$ figli diversi. Considerando il cosiddetto Cross-Over uniforme in cui vengono scelti i singoli bit dall'uno o dall'altro genitore in modo casuale il numero massimo di figli diversi è in questo caso esponenziale, e pari a 2^L-2.

5.2 Applicazione per la ricostruzione delle tracce

Un algoritmo genetico può essere utilizzato per minimizzare l'errore quadratico medio degli scarti dei tempi di hit (8), analogamente a quanto si fa nella ricostruzione OPNEMO. Gli individui che compongono la popolazione sono formati dalle stringhe di bit che codificano i parametri della traccia x_0, y_0, z_0, θ, e ϕ, come rappresentato in Figura 84

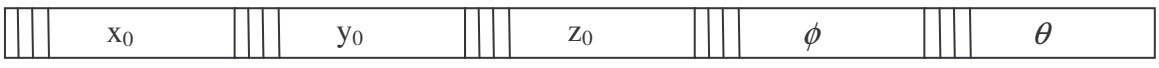

Figura 84 - Codifica dei parametri di traccia in un "cromosoma".

Posto lo schema di principio di un algoritmo genetico generico di Figura 80, i parametri liberi su cui si può agire per sperimentare ed ottimizzare la procedura sono molti. A priori è stata fissata la codifica delle variabili di traccia x_0, y_0, z_0, θ, e ϕ in parole binarie con una risoluzione (Least Significant Bit: LSB) sufficientemente piccola in modo che l'errore di quantizzazione non influisca sull'accuratezza angolare. Scegliendo parole di 16 bit il LSB per θ e ϕ risulta pari a

$$LSB_\theta = \frac{180}{2^{16}-1} = 0.00274662°$$
$$LSB_\phi = \frac{360}{2^{16}-1} = 0.00549325°$$

Per quanto riguarda la posizione iniziale, poiché il volume netto occupato dai PMT ha le dimensioni (±570 m, ±570 m, ±340 m) intorno al centro bisogna ricoprire 1140 m in x e y e 680 m in z, più almeno tre lunghezze di attenuazione dei fotoni. Prendendo L_a=75 m, cioè la lunghezza di attenuazione dell'acqua pura, si ha che le coordinate iniziali possono assumere valori in un intervallo di circa 1600 m. Una parola di 16 bit avrà quindi $LSB_{x,y,z}$ = 2.5 cm, più che sufficiente per l'accuratezza sulla misura dei tempi (≅ 0.083 ns).

La trasformazione dal valore binario al valore in metri delle coordinate è:

$$x_0[m] = X_0\, LSB_x - x_C$$
$$y_0[m] = Y_0\, LSB_y - y_C$$
$$z_0[m] = Z_0\, LSB_z - z_C$$

dove $x_C = y_C = z_C$ = 819.1875 m sono le posizioni del centro dell'apparato rispetto al bordo.

Sono state valutate le prestazioni dell'algoritmo al variare di:

- dimensione della popolazione
- tipo di cross-over
- probabilità di mutazione.

Si è osservato a parità di altre condizioni, un miglioramento delle prestazioni per alti valori di mutazione; questo è dovuto al fenomeno della convergenza evolutiva, per cui dopo molte generazioni vi sono individui identici. Ciò causa la effettiva diminuzione della popolazione efficace. Per ovviare a questo inconveniente è stato introdotto inizialmente nell'algoritmo un controllo di esistenza di cloni, che in caso affermativo ne sostituisce uno con un nuovo individuo casuale. Tuttavia il carico computazionale di questo controllo è molto gravoso ($o(n^2)$). Lo stesso effetto benefico è stato ottenuto tramite un meccanismo di rinnovo parziale della popolazione, cioè ad ogni generazione i peggiori N_{WOR} individui vengono

sostituiti con elementi casuali, (anche N_{WOR} è un parametro da ottimizzare), e questo metodo ha dato buoni risultati con un contenuto carico computazionale.

I risultati delle simulazioni con gli algoritmi genetici sono descritti nel §5.3.

5.3 Implementazione della ricostruzione con Algoritmi Genetici

La popolazione iniziale dell'algoritmo genetico è stata generata a partire dagli angoli ottenuti dalla procedura di prefit, con una distribuzione aleatoria di deviazione standard pari a 200 m per le coordinate e 5° per gli angoli.

Sono state effettuate delle prove di ricostruzione al variare dei parametri caratteristici dell'algoritmo genetico, per trovare una configurazione ottimale. Per motivi di tempo di elaborazione sono stati testati solo dei tagli nello spazio dei parametri, e la configurazione scelta corrisponde ai valori dei parametri per cui si ha una prestazione migliore; in teoria dovrebbero essere fatti variare tutti i parametri contemporaneamente per avere un'ottimizzazione globale. D'altra parte si può ipotizzare che per qualunque valore di probabilità di cross-over (CO) o di mutazione le performance migliorino al crescere del numero di individui.

La grandezza indicativa della prestazione è la mediana dell'errore angolare, per dècade di energia.

Nelle figure seguenti sono riportati i risultati della ricostruzione al variare della dimensione della popolazione, per i tre tipi di CO esaminati. In Figura 85 il CO ad un taglio, in Figura 86 il CO a 2 tagli e in Figura 87 il CO uniforme. La dimensione della popolazione è stata fatta variare da 200 a 400 individui.

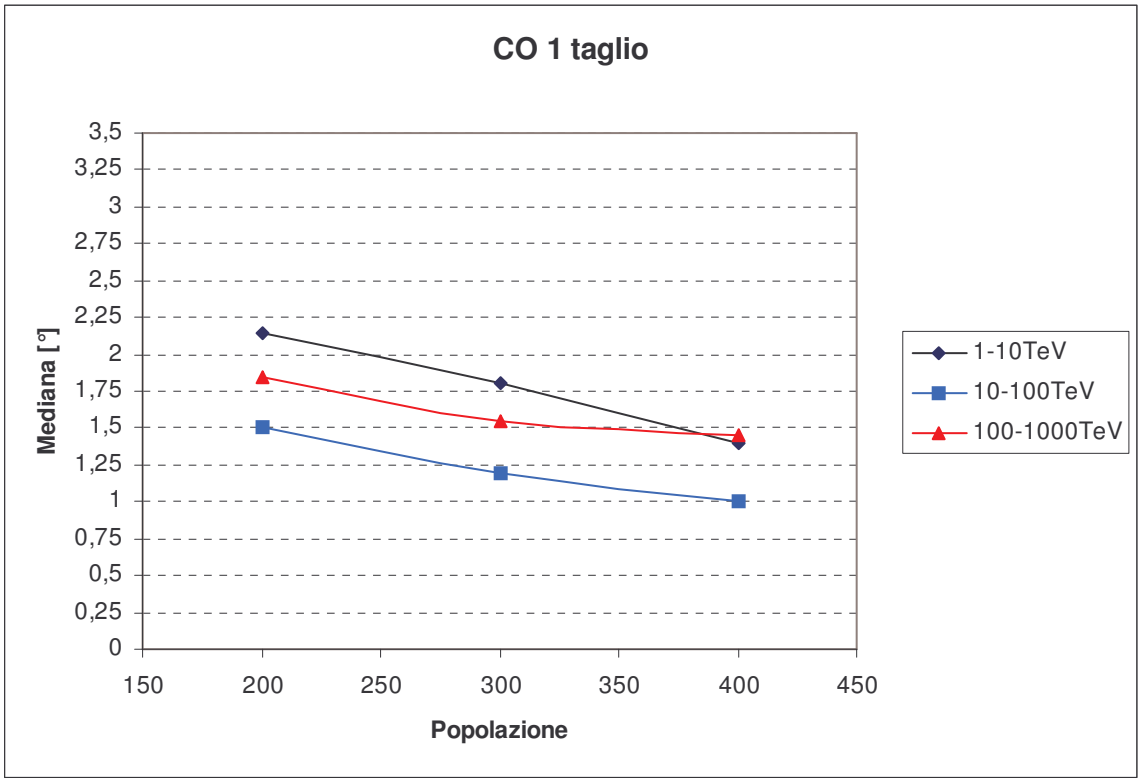

Figura 85 – Risultati della ricostruzione al variare della dimensione della popolazione, con il cross-over ad un taglio.

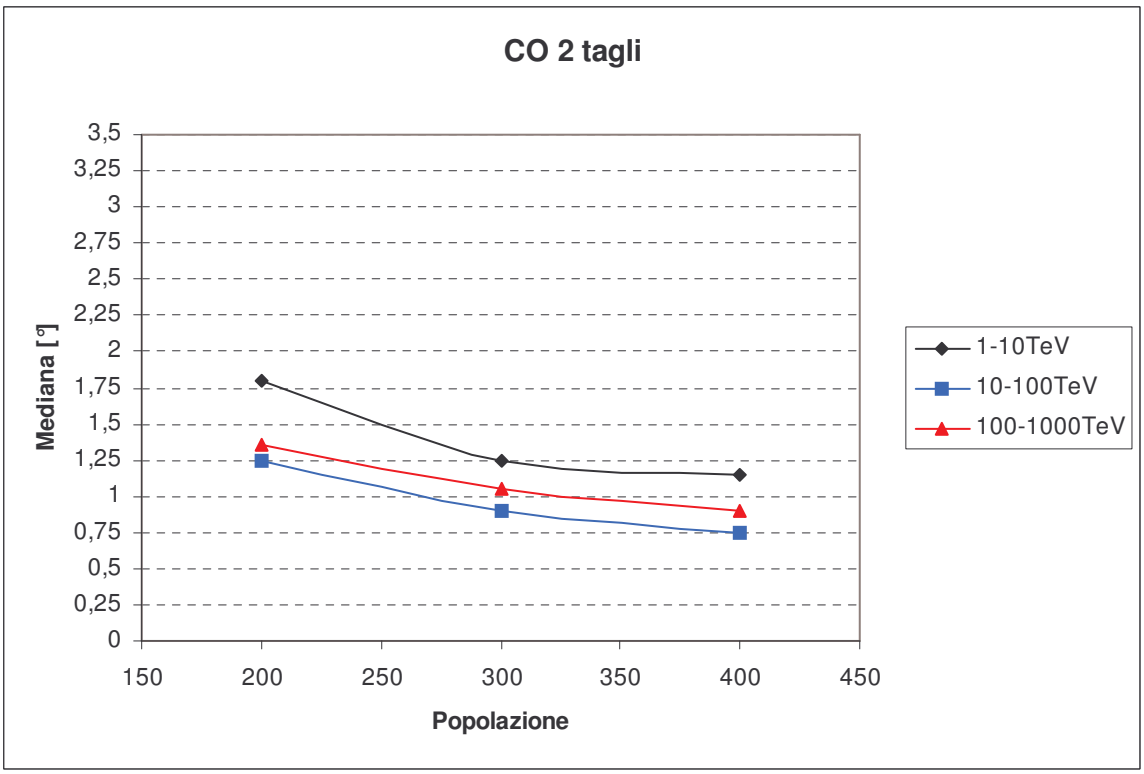

Figura 86 - Risultati della ricostruzione al variare della dimensione della popolazione, con il cross-over a due tagli.

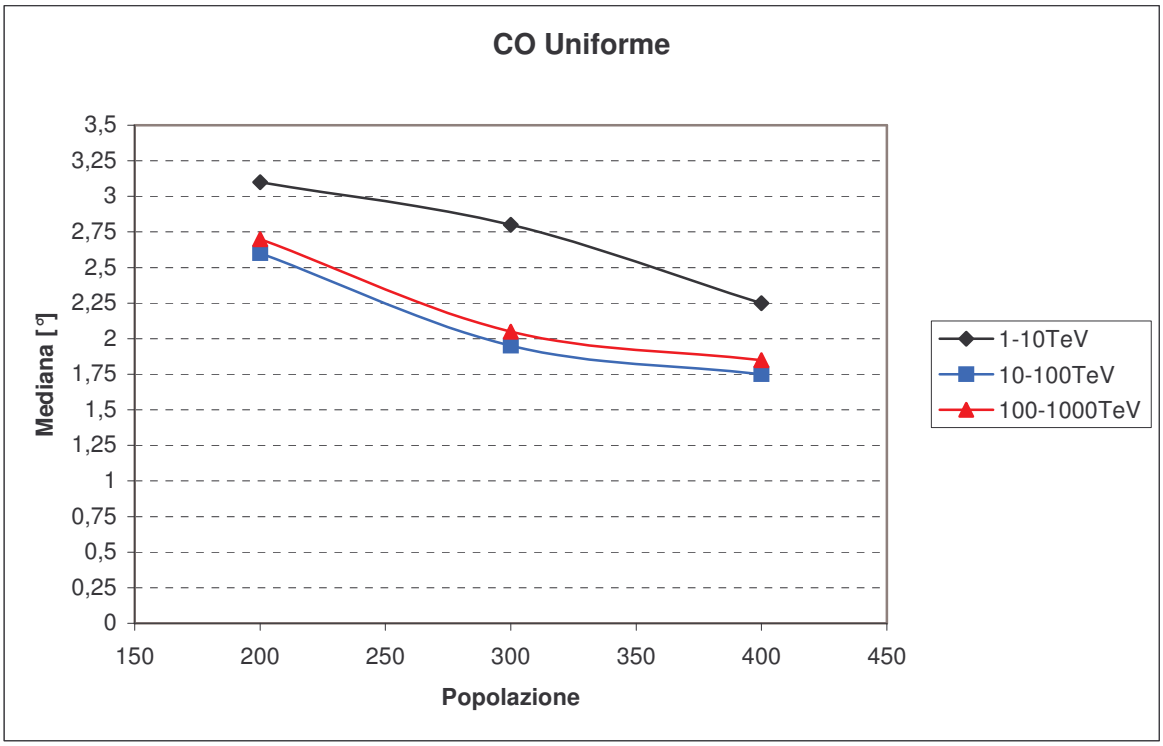

Figura 87 Risultati della ricostruzione al variare della dimensione della popolazione, con il cross-over uniforme.

Scelgo quindi di utilizzare sempre il CO a 2 tagli. Nelle prove successive, inoltre, utilizzerò un numero di individui nella popolazione pari a 200, ipotizzando che le performance migliorino comunque all'aumentare della popolazione.

Nella Figura 88 è riportata la bontà della ricostruzione al variare della probabilità di cross-over.

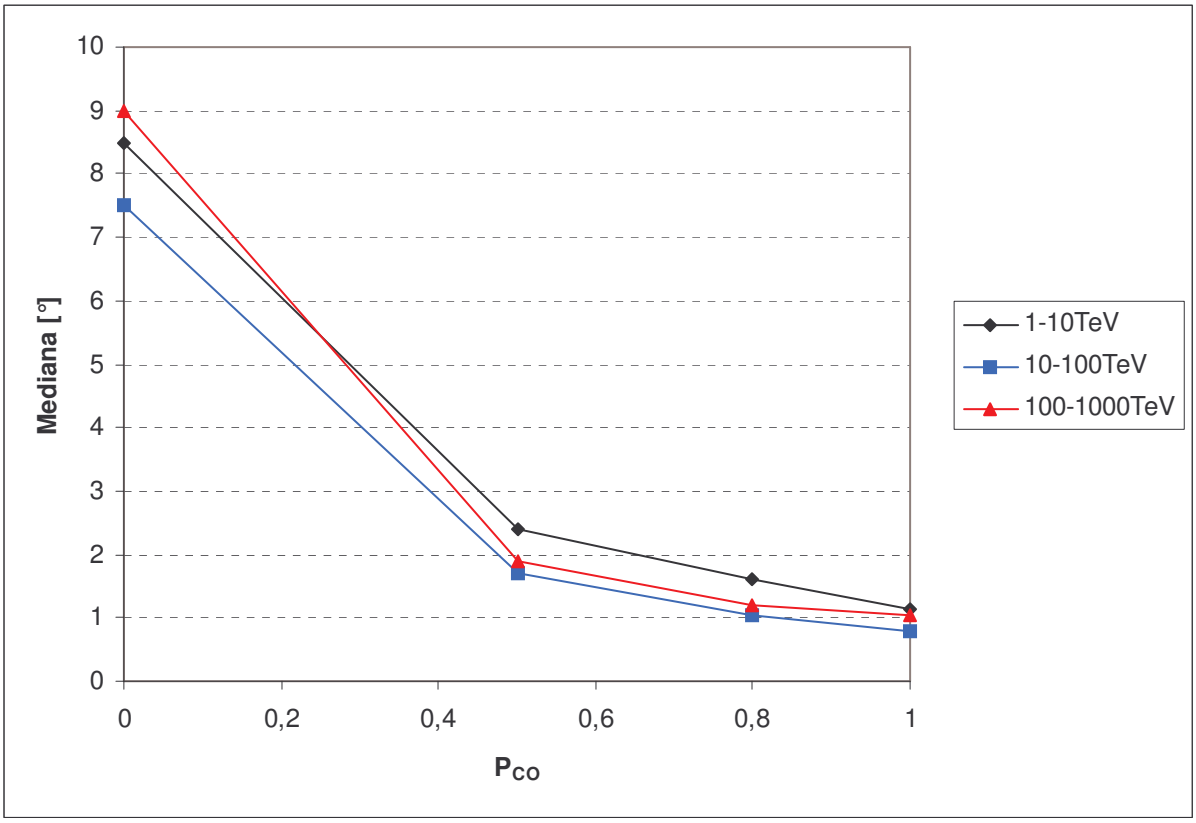

Figura 88 – Accuratezza di ricostruzione al variare della probabilità di cross-over.

Si vede che la migliore performance si ha con probabilità unitaria, d'altra parte in questa implementazione dell'algoritmo genetico il 2% dei migliori individui viene sempre trasmesso alla generazione successiva, per non perdere mai i risultati raggiunti. Il cross over viene applicato a tutti gli altri individui selezionati; è probabilmente per questo motivo che la probabilità di cross over ottima tende all'unità.

Per le successive prove è stata utilizzata una $P_{CO} = 1$.

Nella Figura 89 è riportata l'accuratezza della ricostruzione in funzione della probabilità di mutazione; si nota un miglioramento all'aumentare della mutazione. Ma per mutazioni superiori al 50% l'algoritmo genetico assomiglia sempre di più ad una ricerca randomica. Si è provato pertanto ad inserire nell'algoritmo un rinnovo della popolazione in modo casuale, sostituendo una percentuale dei peggiori individui. Il risultato di questa operazione è riportato in Figura 90 al variare della percentuale di individui rinnovati. Si vede che è stato trovato un minimo dell'errore angolare intorno al 10% di rinnovo, e poco variabile con la probabilità di mutazione.

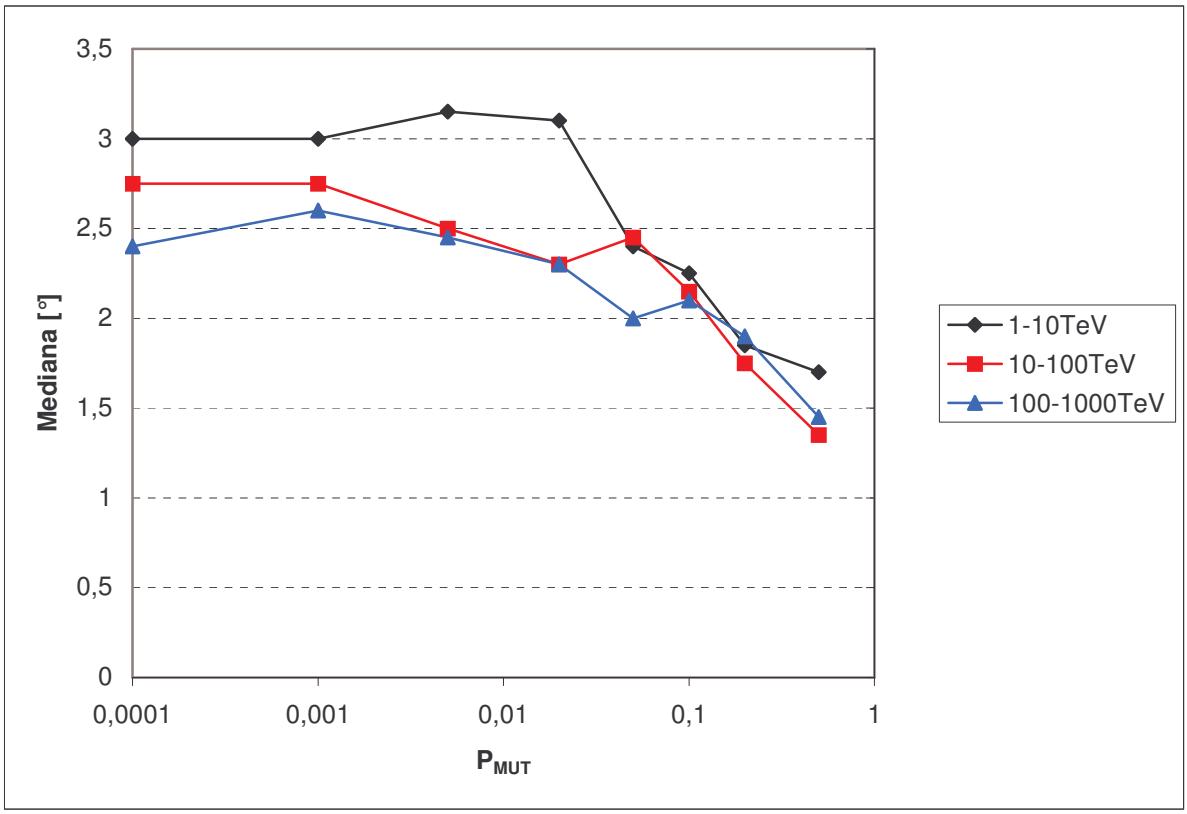

Figura 89 – Accuratezza della ricostruzione al variare della probabilità di mutazione.

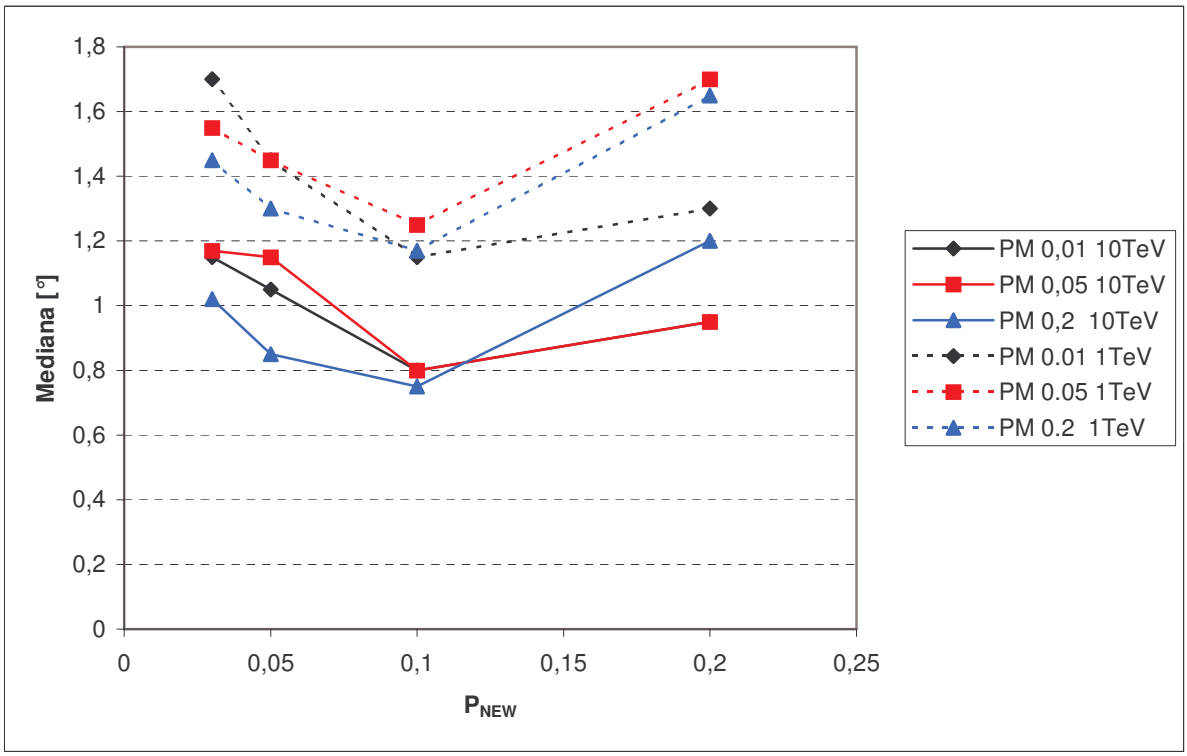

Figura 90 – Accuratezza di ricostruzione minima (linee continue, 10 - 100 TeV) e massima (linee tratteggiate, 1-10 TeV) in funzione della percentuale di rinnovo e della probabilità di mutazione.

La configurazione scelta per la ricostruzione è quella con il Cross Over a due tagli, con un rinnovo della popolazione pari al 10% ad ogni generazione, con probabilità di cross-over pari a 1, e con probabilità di mutazione del 2%.

Invece di aumentare molto il numero di individui nella popolazione, si è preferito ripetere la ricostruzione più volte per ogni evento, mantenendo la prova con più basso valore di χ^2 raggiunto.

Le prestazioni di ricostruzione ottenute utilizzando lo stesso campione di eventi del Cap. 3, sono riassunte nei grafici seguenti.

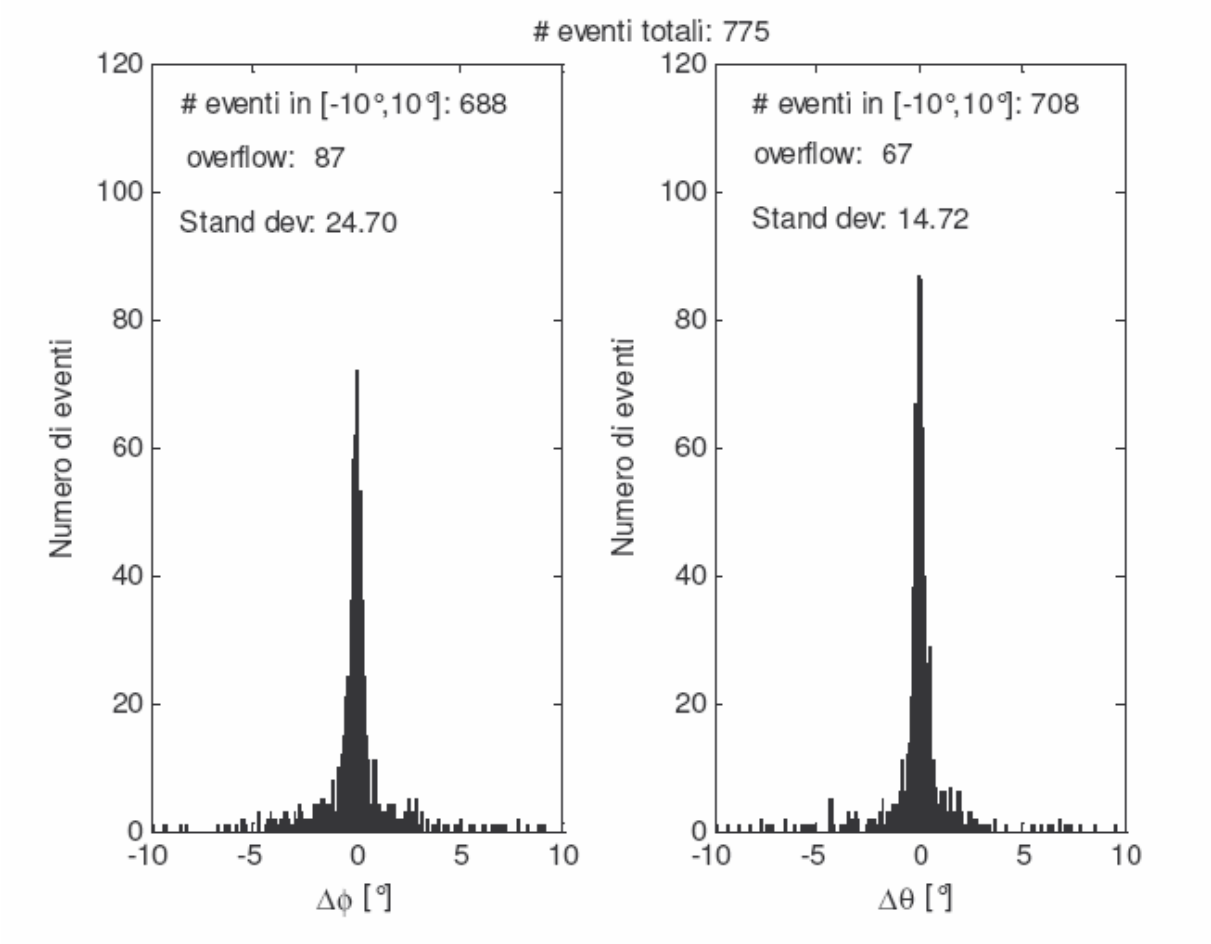

Figura 91 – Istogrammi delle differenze tra gli angoli ϕ e θ della traccia "vera" e quelli della traccia ricostruita dall'algoritmo genetico.

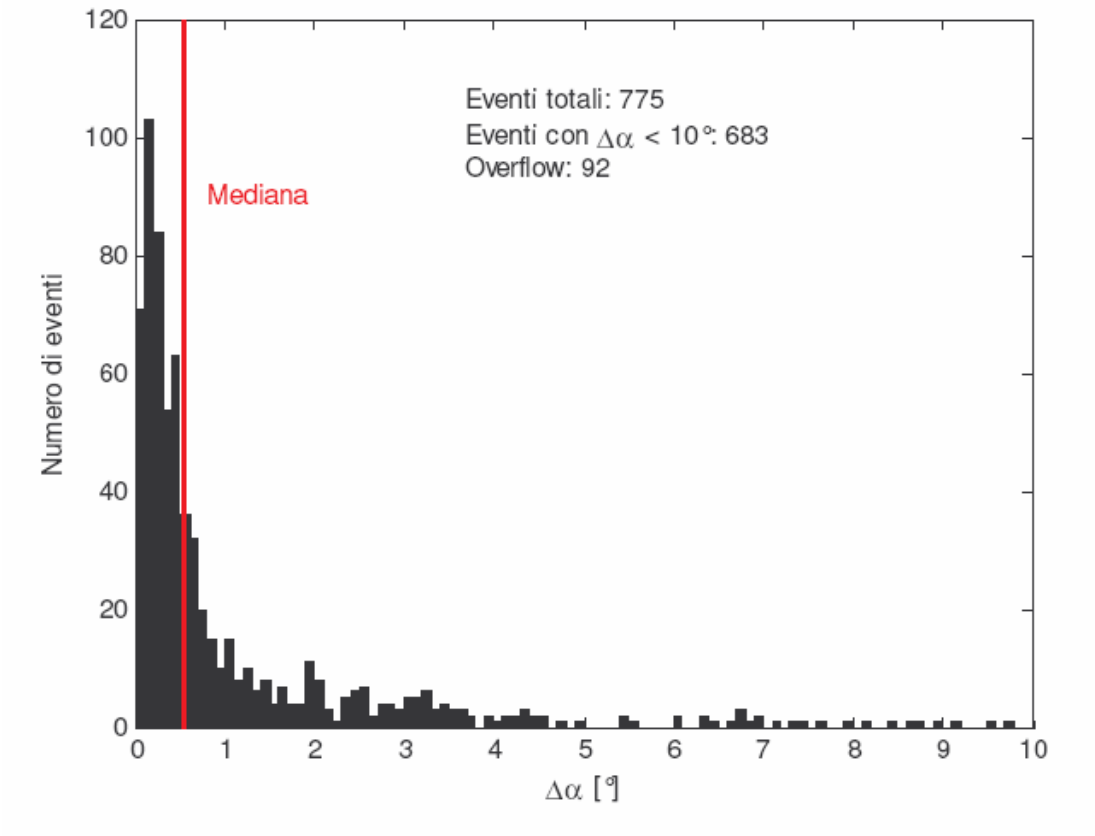

Figura 92 – Istogramma dell'angolo nello spazio tra la traccia "vera" e quella ricostruita.

Nella Figura 91 sono disegnati gli istogrammi delle differenze angolari in ϕ e θ, mentre in Figura 92 è riportato l'angolo nello spazio ($\Delta\alpha$) tra la traccia vera e quella ricostruita. Confrontandolo con quello della ricostruzione classica di Figura 52 si vede che le performance sono molto simili. La ricostruzione classica dà una mediana totale pari a $0.5^{\circ}\,^{+0.1}_{-0.07}$, mente la ricostruzione con l'algoritmo genetico dà una mediana totale pari a $0.53^{\circ}\,^{+0.08}_{-0.06}$.

In Figura 93 è riportato lo scatter plot dell'errore angolare rispetto alle coordinate del vertice della traccia, e in Figura 94 rispetto a θ e ϕ.

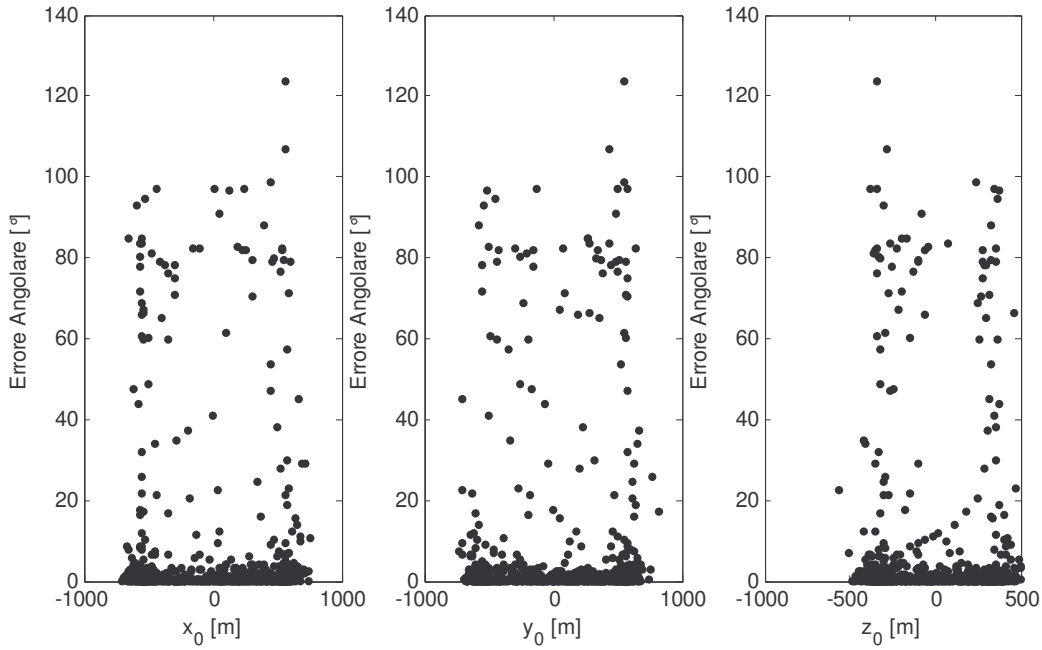

Figura 93 – Errore angolare vs coordinate dei vertici della traccia.

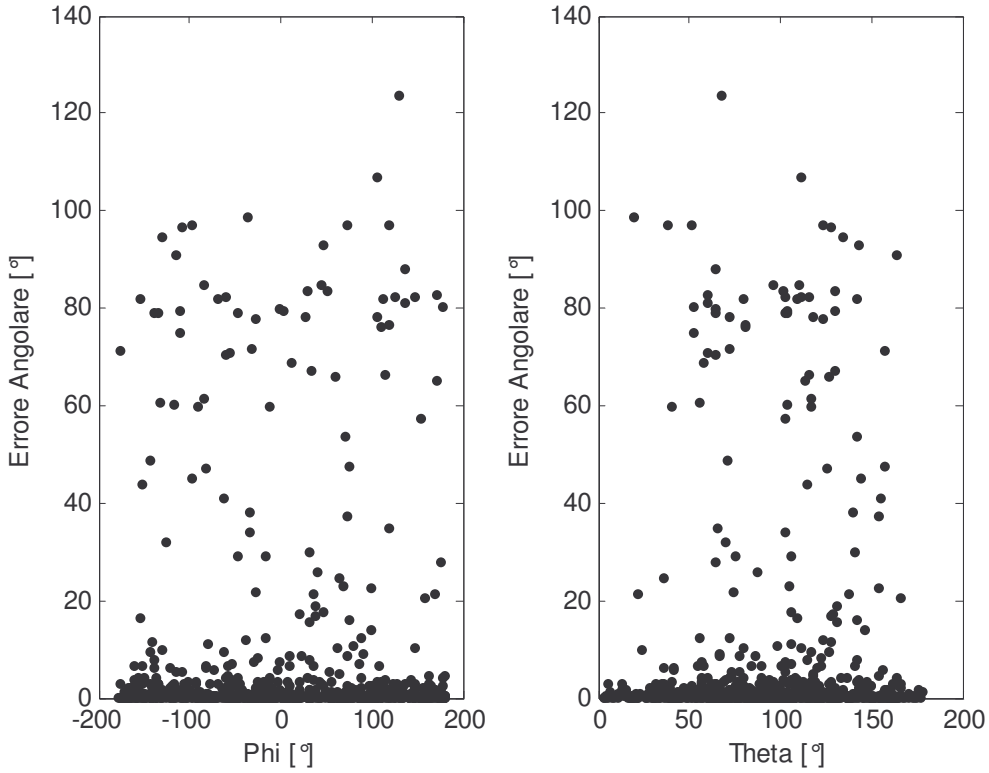

Figura 94 – Errore angolare vs θ e ϕ.

L’errore angolare rispetto all’energia è riportato in Figura 95 (scatter plot) e in Figura 96 (istogrammi delle standard deviation e delle mediane)

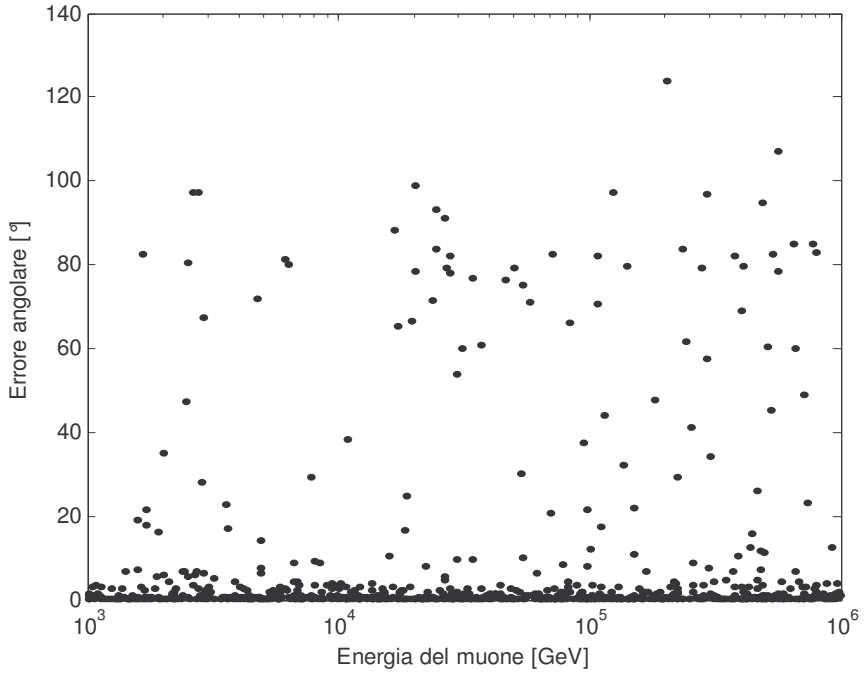

Figura 95 – Errore angolare in funzione dell’energia (scatter plot).

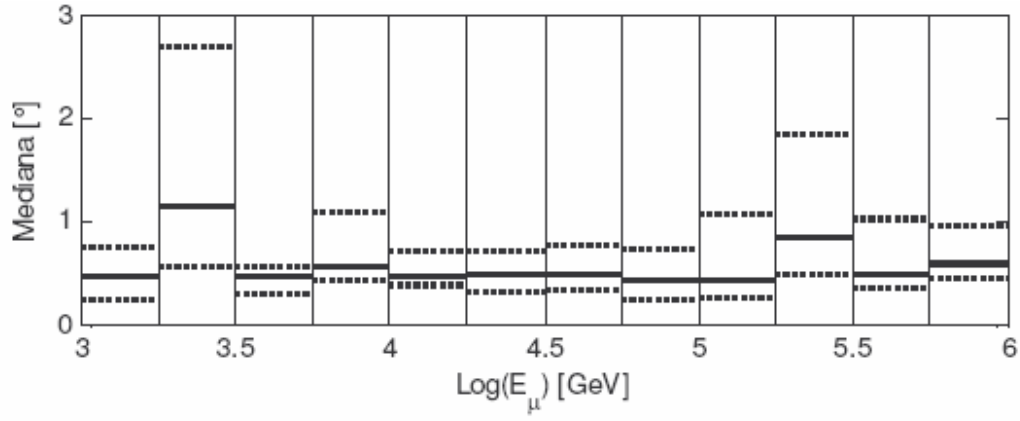

Figura 96 – Deviazione Standard e mediana della differenza angolare in funzione dell’energia (in linea tratteggiata l'errore a 1 σ).

In Figura 97 è riportato l’errore angolare in funzione del numero di hit, in Figura 98 sono riportati gli istogrammi (standard deviation e mediana), insieme alla mediana totale degli eventi con N_{hit} superiore al valore corrente (linea rossa).

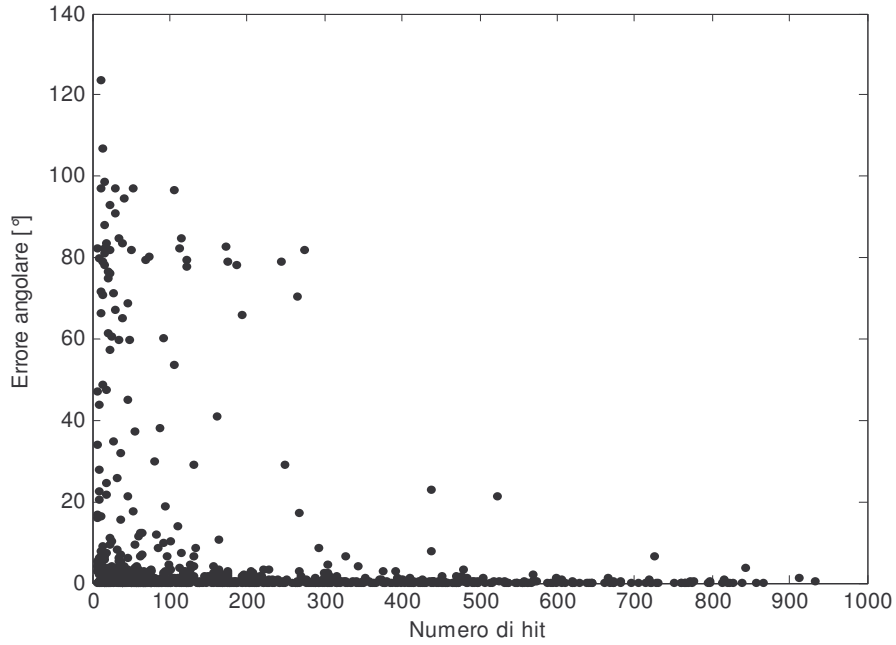

Figura 97 – Errore angolare rispetto al numero di hit.

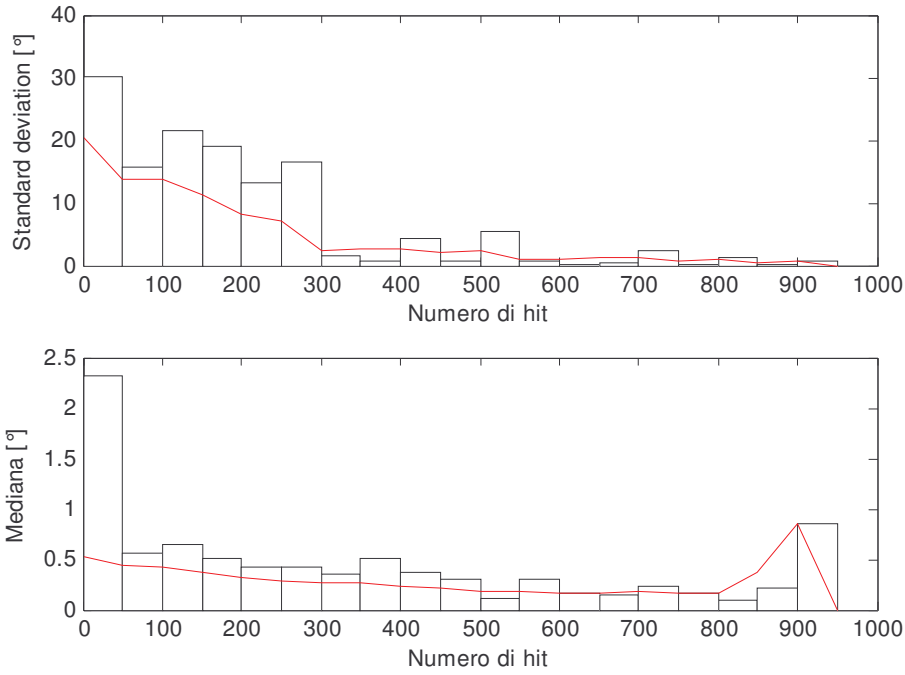

Figura 98 – Istogrammi della standard deviation e della mediana in funzione del numero di hit, e valori totali per numero di hit superiore al valore corrente (linea rossa).

In Figura 99 è riportata la mediana dell'errore angolare in funzione del χ^2. Applicando lo stesso filtro sul χ^2 scelto nel §3.6,si hanno le performance di Figura 100.

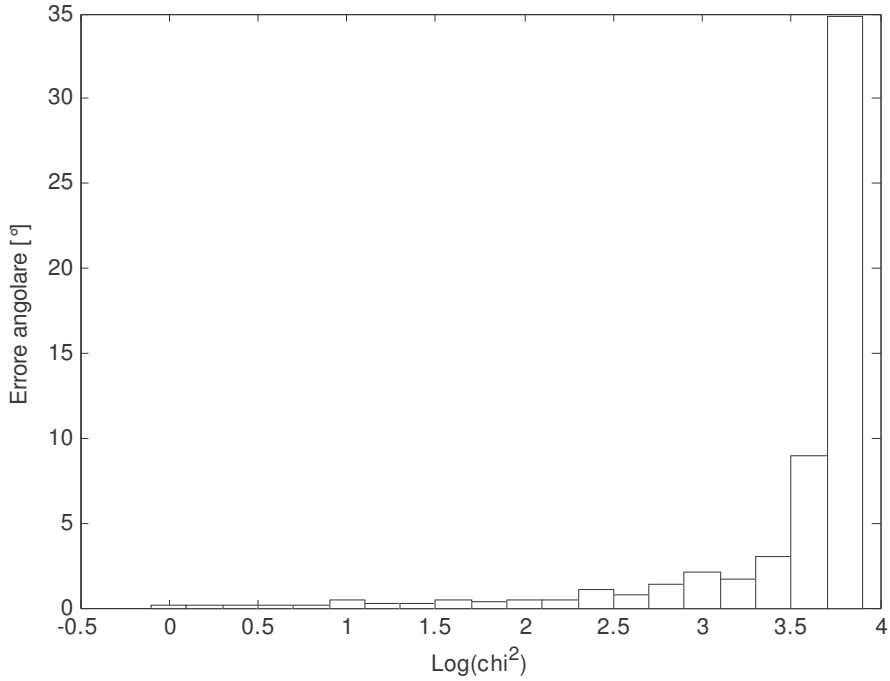

Figura 99 – Errore angolare (mediana) in funzione del χ^2.

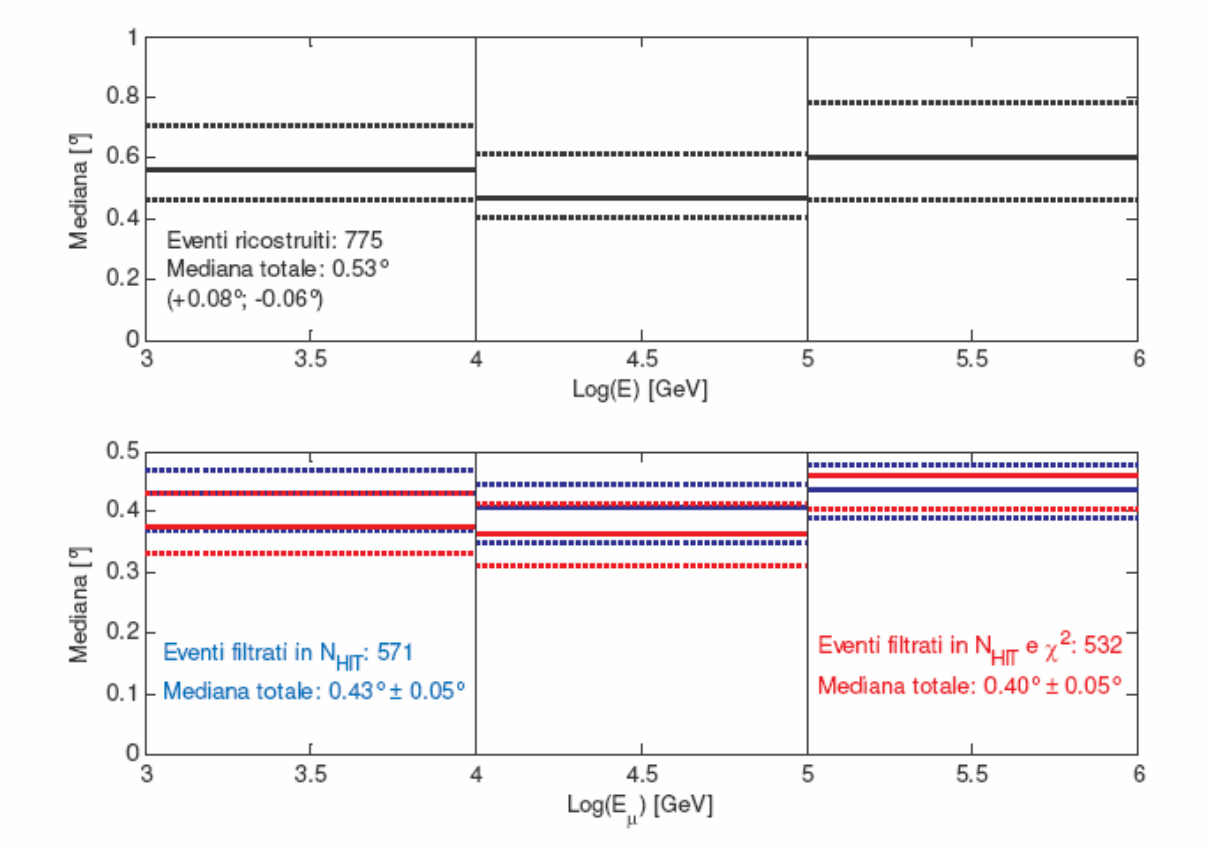

Figura 100 – Mediana dell'errore angolare in funzione dell'energia, prima e dopo la selezione.

La mediana totale dell'errore per tutti gli eventi ricostruiti ($N_{hit}>5$) è quindi $0.53°^{+0.08}_{-0.06}$.
Con la selezione sul numero di hit la mediana dell'errore totale è 0.43° ± 0.05; con la selezione sul numero di hit e sul χ^2 la mediana dell'errore è 0.40° ± 0.05.

Efficienza di ricostruzione

L'area efficace prima della selezione è pari a $A_{Eff} = A_G \cdot 775/1000 = 2.77$ km^2; dopo la selezione l'area efficace risulta $A_{Eff} = A_G \cdot 532/1000 = 1.9$ km^2, con una mediana dell'errore pari a 0.40° ± 0.05°.

In Figura 101 è riportata l'area efficace (rapportata all'area geometrica) al variare dell'energia.

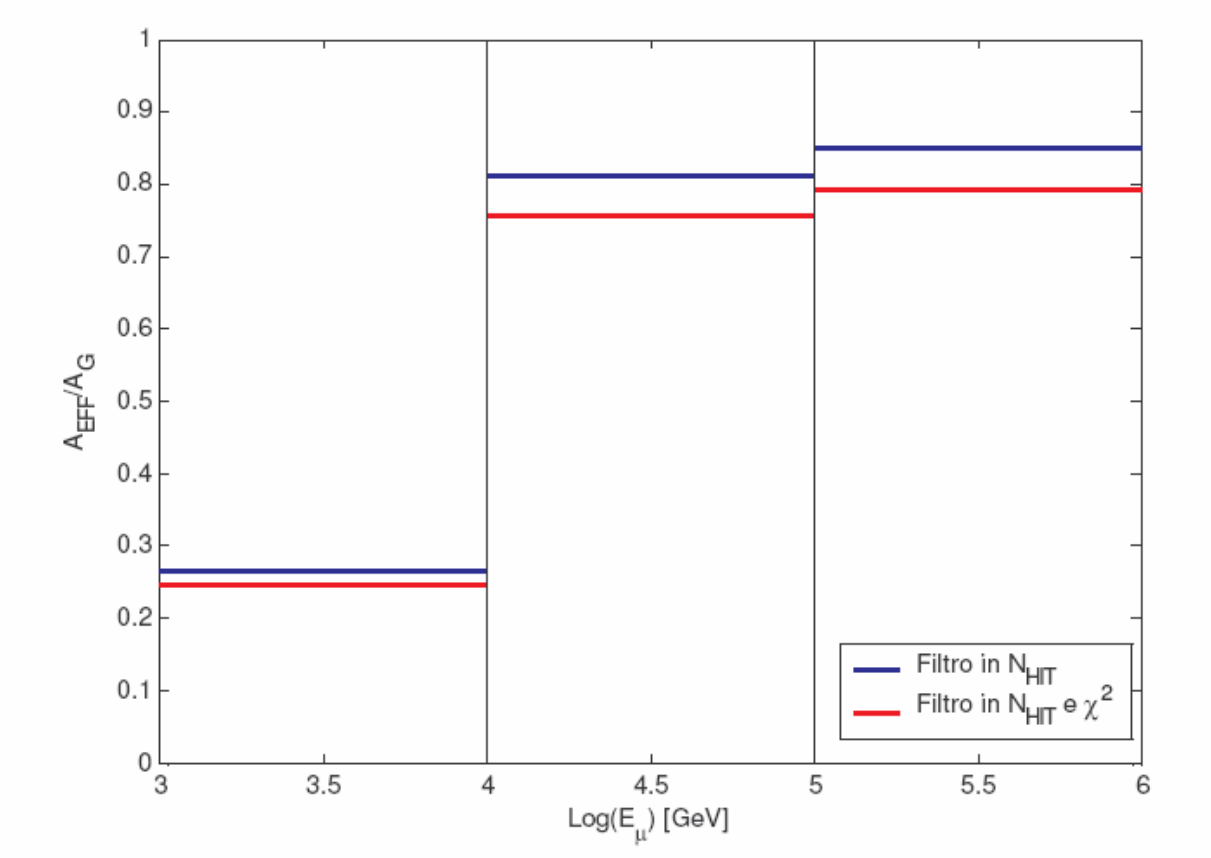

Figura 101 - Area efficace normalizzata all'area geometrica in funzione dell'energia.